Getting Started with Isabelle*

Jacques Fleuriot

October 22, 2013

1 Starting and Exiting Isabelle

Under DICE, you can start the proof assistant Isabelle, with its *Proof General* front end, by entering the following in a terminal window:

```
isabelle emacs
```

The `emacs` here refers to the *emacs* text editor. Emacs is a powerful extensible editor, and Proof General is built to run within emacs. While emacs can be used in a point-and-click mode with the help of the menus, there are number of key bindings that most emacs users find useful. To learn more about emacs, launch the *emacs tutorial* from the top of the *Help* menu.

Note that if you run into difficulties trying to get the standard DICE version of Isabelle to work, you may set up your bash shell environment to use an alternative installation. To do this, first type the following in a terminal window:

```
source /afs/inf.ed.ac.uk/group/dreamers/public/software/Isabelle2013/source_me.sh
```

and then proceed as before by typing:

```
isabelle emacs
```

Isabelle itself runs as a sub-process, automatically started from within Proof General when you first request Isabelle proof script to be processed.

The theorem prover has a number of object-level logics. By default it is started for the HOL (Higher-Order Logic) object logic – the one that we use in this course. If you wish to use a different one, you should select it using the Isabelle menu before issuing any Isabelle proof script processing commands.

To exit from Proof General, select *Exit Isabelle* from the `Isabelle` menu. Usually you will then get a pop-up window asking about killing the Isabelle sub-processes. Answer *yes*.

*This short guide is about using Isabelle with the Proof General interface. For its use with the jEdit interface, please consult the official Isabelle webpage.

2 Isabelle Theories

Definitions, theorem statements, proofs and comments are grouped together in Isabelle into *theories*. Each theory is stored in a file with extension `.thy`. To work on new or existing theory in Proof General, load the theory file into emacs buffer using the *Open File* command on the *File* menu. Proof General sets up emacs to recognise the `.thy` filename extension and switch the buffer into a mode (*Isar script* mode) for editing and processing theory files. *Isar* is the name of the language Isabelle theories are written in. Isar has many features supporting the writing of *structured proofs*, proof descriptions where the flow of the logical arguments is readily visible and a formal approximation of how proofs are presented in maths textbooks. In this course, we don't have time to cover these structured proof features of Isar, and we stick with a simpler more-operational proof style.

By default, when Proof General starts up, you are presented with an empty theory file `Scratch.thy`. In general, multiple theory files can be loaded into Proof General, but only one is *active* at a time, ready to be processed by Isabelle.

The general structure of a theory is shown by the following example.

```
theory Prop
imports Main

begin

theorem A: "A ⟶ A ∨ B"
apply (rule impI)
apply (rule disjI1)
apply assumption
done

theorem B: "A ∨ B ⟶ B ∨ A"
apply (rule impI)
apply (erule disjE)
apply (rule disjI2)
apply assumption
apply (rule disjI1)
apply assumption
done

end
```

Here `Prop` is the name of a theory. In general, a theory with name *Name* must be stored in file *Name*`.thy`. For example, this theory `Prop` should be saved in a file `Prop.thy`.

Theories typically build upon existing theories Here we load the parent theory `Main`, which gives us our inference rules for propositional logic. Note that it also loads up the theories of more expressive logics. We will not use them until later in the course!

Between the `begin` and `end` keywords we enter theory elements such as definitions, theorems and proofs.

3 Proof General

The Proof General interface to Isabelle allows us to walk through the proof in a way that is easy to manipulate.

At any one time, Isabelle has processed some portion of the active theory file. The processed portion is highlighted with a blue background and is made read only. The unprocessed portion below has a white background and is freely editable. While Isabelle is processing some command, the background of the command text it set to pink. This might well be unnoticed when Isabelle processes simple commands in a small fraction of a second.

Several of the Proof General buttons (below the menus at the top of the window) control the processed portion. These include:

- **Next**: process next proof command
- **Undo**: undo the previous proof command
- **Goto**: process or undo everything upto the cursor position
- **Use**: process whole buffer
- **Retract**: Retract (undo) whole buffer

Hover the mouse over each button to see a short description of what it does.

4 Using Isabelle and Unicode Tokens

Isabelle can (should!) be used in unicode token mode, which will give you math-like symbols. For example, typing ==> will give you the symbol $\Longrightarrow$.

Table 1 shows most common symbols you'll encounter The *key-strokes* column indicates what to type to enter the symbol. The *token* column shows the standard character sequence used for representing the symbol when the buffer is saved to a file. Isabelle/HOL does not use the $\top$ and $\bot$ symbols for true and false. Rather it uses `True` and `False`.

A list of all symbols can be displayed by selecting the menu option *Tokens > List Tokens*. If you move the mouse over a character and wait for a short while, emacs will display some information about it at the bottom of the

symbol	key-strokes	token
⟦	`[ |`	`\<lbrakk>`
⟧	`| ]`	`\<rbrakk>`
⟹	`= = >`	`\<Longrightarrow>`
∧	`/ \`	`\<or>`
∨	`\ /`	`\<and>`
¬	`\not`	`\<not>`
⟶	`- - >`	`\<longrightarrow>`
⟷	`< - - >`	`\<longleftrightarrow>`
∀	`ALL`	`\<forall>`
∃	`EX`	`\<exists>`

Table 1: Unicode Tokens

window, including how to type the symbol in directly. Alternatively, you may select the mathematical symbol that you wish to insert at the current position in your proof script by selecting the *Maths* menu and then the appropriate command from one of the sub-menus.

5 Continuing

After reading and following these instructions, and consulting the slides from the first two weeks of lectures, you should be ready to attempt some propositional logic proofs in Isabelle.

By default, Isabelle in the background will try to both prove and find a counter-example to a theorem you are attempting to prove. The messages from these attempts can be distracting when first using Isabelle. To turn them off, de-select *Auto Solve* and *Auto Quickcheck* on the *Isabelle > Settings* sub-menu.

Some suggestions for further reading are :

- An *overview of Isabelle* at
  `http://www.cl.cam.ac.uk/research/hvg/Isabelle/overview.html`
- The *Isabelle FAQ* available at
  `https://isabelle.in.tum.de/community/FAQ`
- An *Isabelle 'Cheat Sheet'*

  prepared by Jeremy Avigad at CMU. Available at
  `http://www.phil.cmu.edu/~avigad/formal/FormalCheatSheet.pdf`.

  This sumarises Isabelle's proof methods and logic rules. It is also available at
  `http://www.inf.ed.ac.uk/teaching/courses/ar/isabelle/FormalCheatSheet.pdf`

- The *Isabelle documentation*,

  including tutorials and reference manuals. For the current version (Isabelle 2013), access the documentation at:

  `http://www.cl.cam.ac.uk/research/hvg/Isabelle/documentation.html`

  Warning: virtually all of this goes into Isabelle in much more detail than is needed for this course, right from the start. From the Isabelle home page you can also access downloads of Isabelle, should you wish to install your a personal copy on your own computer.

- The *Proof General documentation*

  available from the menu *Proof General* > *Help* > *Info*.

- The *i3p alternative front end*.

  This is an alternative to Proof General (Caution: compatibility with Isabelle 2013 is unclear). Visit
  `http://www-pu.informatik.uni-tuebingen.de/i3p/`.